DETAILED ACTION

1. This action is in response to the amendment filed on 01/05/2010.

2. The objections to the specification are withdrawn in view of applicant's remarks/arguments.

3. The rejection under 35 U.S.C. 102(b) as being anticipated by Wang (US 2003/0088860) to claims 1-30 is moot in view of new ground(s) of rejection.

4. Claims 1, 8, 18 and 25 have been amended.

5. Claims 5-7, 12-14 and 22-24 have been canceled.

6. Claims 1-4, 8-11, 15-21 and 25-30 are pending.

Information Disclosure Statement

7. The information disclosure statement filed on 4/21/06 fails to comply with the provisions of 37 CFR 1.97, 1.98 and MPEP § 609 because full citation was not given for NPL document. It has been placed in the application file, but the information referred to therein has not been considered as to the merits. Applicant is advised that the date of any re-submission of any item of information contained in this information disclosure statement or the submission of any missing element(s) will be the date of submission for purposes of determining compliance with the requirements based on the time of filing the statement, including all certification requirements for statements under 37 CFR 1.97(e). See MPEP § 609.05(a).

8. The information disclosure statement (IDS) submitted on 4/9/08 is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

Response to Amendment

Claim Rejections - 35 USC § 101

9. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

10. Claims 18-21 and 25-30 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

11. Claim 18 reciting a "compiler" fails to fall in a statutory category, such as a method, system, or article of manufacture. As per claims 19-21, these claims are rejected for failing to cure the deficiencies of the above rejected base claim 18.

12. Claim 25 reciting a "translator" fails to fall in a statutory category, such as a method, system, or article of manufacture. As per claims 26-30, these claims are rejected for failing to cure the deficiencies of the above rejected base claim 25.

Claim Rejections - 35 USC § 103

13. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

14. Claims 1-4, 8-11, 15-21 and 25-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wang (US 2003/0088860) in view of Goebel (U.S. 5,901,316).

Per Claim 1:

Wang teaches compiling source code into source binary code for a first computing platform ("…In accordance with the present invention, an optimizing compiler adds annotation information (compiler annotation) to an executable binary code file. …" in par. 0009); and generating an annotation section associated with the source binary code, the annotation section comprising an annotation for a scope, the scope comprising at least one block of the source binary code having at least one attribute to aid a translator optimization ("…Compiler annotation provides information useful for binary translators such that a binary translator does not have to use a heuristic approach to translate binary code. Compiler annotation identifies such information as function boundaries, split functions, jump table information, function addresses, and code labels. The compiler annotation can be used by a binary translator when translating a source binary code to a target binary code. The target binary code optionally includes new compiler annotation. …" in par. 0009), wherein the at least one attribute comprises at least one of information associated with local variable assignment within the at least one block ("… Function address assignment ID record 412 can be used to identify function addresses and other code labels which are used by, for example, `sethi`/`or` instructions, to generate code addresses. Code addresses used in these instructions need to be updated when an address of code is changed

during binary transformation. Function address assignment information is generated, for example, when an address of a function is taken by the executable binary code. …" in par. 0056; code address of a function that contains the local variable assignment is contained in Function address assignment ID record) and information associated with volatile variable access within the at least one block ("…Volatile load ID record 418 can be used to identify the address of a volatile load. A volatile memory reference must not be removed or re-ordered with respect to other volatile memory references. …" in par. 0059; volatile memory reference is interpreted as the information associated with volatile variable access). Wang does not explicitly teach wherein the at least one attribute comprises at least one of information associated with register spilling and restoring instructions within the at least one block.

However, Goebel teaches wherein the at least one attribute comprises at least one of information associated with register spilling and restoring instructions within the at least one block ("…As can be appreciated by one skilled in the art, the code optimizer 52 will operate, according to the present invention, in a manner not greatly unlike prior art code optimizers. The code optimizer 52 will simply have available the float spill cache 22 as potential locations to which to spill the single precision registers 14, in addition to the conventional memory stack 18. The question of which of the spill slots 24 may be available at any given instant is tracked according to the conventional prior art methods for keeping track of any and all resources which are instantly available. When the code optimizer 52 determines, according to such prior art methods, that it is appropriate to spill one of the single precision registers 14 to a spill slot 24, then the instructions previously described in relation to FIG. 2 for accomplishing the inventive spill operation 26 are inserted. …" in column 9, lines 39-55).

It would have been obvious to one having ordinary skill in the computer art at the time of the invention was made to modify the method disclosed by Wang to include wherein the at least one attribute comprises at least one of information associated with register spilling and restoring instructions within the at least one block using the teaching of Goebel. The modification would be obvious because one of ordinary skill in the art would be motivated to utilize registers in an efficient manner (Goebel, column 3, lines 10-15).

Per Claim 2:

Wang further teaches wherein if the scope comprises a plurality of blocks, the blocks have consecutive addresses with each other and have the at least one attribute in common (par. 0048).

Per Claim 3:

Wang further teaches wherein the annotation section further comprises a region annotation for a region comprising one or more scope (par. 0047).

Per Claim 4:

Wang further teaches wherein the annotation for the scope further comprises scope addresses, scope size and the at least one attribute, and the annotation for the region further comprises a region annotation pointer, region addresses and region size (par. 0048).

Per Claim 8:

Wang further teaches inputting source binary code for a first computing platform and an annotation section associated with the source binary code ("…In accordance with the present invention, an optimizing compiler adds annotation information (compiler annotation) to an executable binary code file. …" in par. 0009); and translating the source binary code to a target binary code for a second computing platform by utilizing the annotation section, wherein, an annotation section comprises an annotation for a scope, the scope comprising at least one block of the source binary code having at least one attribute to aid a translator optimization ("…Compiler annotation provides information useful for binary translators such that a binary translator does not have to use a heuristic approach to translate binary code. Compiler annotation identifies such information as function boundaries, split functions, jump table information, function addresses, and code labels. The compiler annotation can be used by a binary translator when translating a source binary code to a target binary code. The target binary code optionally includes new compiler annotation. …" in par. 0009), and wherein the at least one attribute comprises at least one of information associated with local variable assignment within the at least one block ("… Function address assignment ID record 412 can be used to identify function addresses and other code labels which are used by, for example, `sethi`/`or` instructions, to generate code addresses. Code addresses used in these instructions need to be updated when an address of code is changed during binary transformation. Function address assignment information is generated, for example, when an address of a function is taken by the executable binary code. …" in par. 0056; code address of a function that contains the local variable assignment is contained in Function address assignment ID record) and information associated with volatile variable access within the at least one block ("…Volatile load ID record 418 can be

used to identify the address of a volatile load. A volatile memory reference must not be removed or re-ordered with respect to other volatile memory references. …" in par. 0059; volatile memory reference is interpreted as the information associated with volatile variable access). Wang does not explicitly teach wherein the at least one attribute comprises at least one of information associated with register spilling and restoring instructions within the at least one block.

However, Goebel teaches wherein the at least one attribute comprises at least one of information associated with register spilling and restoring instructions within the at least one block ("…As can be appreciated by one skilled in the art, the code optimizer 52 will operate, according to the present invention, in a manner not greatly unlike prior art code optimizers. The code optimizer 52 will simply have available the float spill cache 22 as potential locations to which to spill the single precision registers 14, in addition to the conventional memory stack 18. The question of which of the spill slots 24 may be available at any given instant is tracked according to the conventional prior art methods for keeping track of any and all resources which are instantly available. When the code optimizer 52 determines, according to such prior art methods, that it is appropriate to spill one of the single precision registers 14 to a spill slot 24, then the instructions previously described in relation to FIG. 2 for accomplishing the inventive spill operation 26 are inserted. …" in column 9, lines 39-55).

It would have been obvious to one having ordinary skill in the computer art at the time of the invention was made to modify the method disclosed by Wang to include wherein the at least one attribute comprises at least one of information associated with register spilling and restoring instructions within the at least one block using the teaching of Goebel. The modification would

be obvious because one of ordinary skill in the art would be motivated to utilize registers in an efficient manner (Goebel, column 3, lines 10-15).

Per Claim 9:

Wang further teaches wherein if the scope comprises a plurality of blocks, the blocks have consecutive addresses with each other and have the at least one attribute in common (par. 0048).

Per Claim 10:

Wang further teaches wherein the annotation section further comprises an annotation for a region comprising one or more scope (par. 0047).

Per Claim 11:

Wang further teaches wherein the annotation for the scope further comprises scope addresses, scope size and the at least one attribute, and the annotation for the region further comprises a region annotation pointer, region addresses and region size (par. 0048).

Per Claim 15:

Wang further teaches wherein translating the source binary code for the source platform further comprises: generating target intermediate code based upon the source binary code; optimizing the target intermediate code by utilizing the annotation section; and generating the

target binary code for the target platform based upon the optimized target intermediate code (par. 0064).

Per Claim 16:

Wang further teaches wherein optimizing the target intermediate code further comprises: generating an internal representation for the annotation section in response to determining that the internal representation has not been established; reading from the annotation section an attribute associated with a block of the target intermediate code based upon the internal representation; and optimizing the block based upon the read attribute (par. 0064).

Per Claim 17:

Wang further teaches wherein the internal representation is an AVL tree, a node of the AVL tree comprising region addresses and region annotation pointer (par. 0062 and 0064).

Per Claims 18-21:

These are compiler versions of the claimed method discussed above (claims 1-4, respectively), wherein all claim limitations also have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also obvious.

Per Claims 25-30:

These are translator versions of the claimed method discussed above (claims 8-11 and 15-16, respectively), wherein all claim limitations also have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also obvious.

Response to Arguments

15. Applicant's arguments with respect to claims 1-4, 8-11, 15-21 and 25-30 have been considered but are moot in view of the new ground(s) of rejection.

In the remarks, the applicant argues that:

a) Wang fails to teach wherein the at least one attribute comprises at least one of information associated with local variable assignment within the at least one block, and information associated with volatile variable access within the at least one block as recited in claims 1, 8, 18 and 25.

Examiner's response:

a) Wang teaches wherein the at least one attribute comprises at least one of information associated with local variable assignment within the at least one block ("… Function address assignment ID record 412 can be used to identify function addresses and other code labels which are used by, for example, `sethi`/`or` instructions, to generate code addresses. Code addresses used in these instructions need to be updated when an address of code is changed during binary transformation. Function address assignment information is generated, for example, when an address of a function is taken by the executable binary code. …" in par. 0056; code address of a

function that contains the local variable assignment is contained in Function address assignment ID record) and information associated with volatile variable access within the at least one block ("…Volatile load ID record 418 can be used to identify the address of a volatile load. A volatile memory reference must not be removed or re-ordered with respect to other volatile memory references. …" in par. 0059; volatile memory reference is interpreted as the information associated with volatile variable access).

In addition, see the rejection above in par. 14 for rejection to claims 1, 8, 18 and 25.

Conclusion

16. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

17. Any inquiry concerning this communication from the examiner should be directed to Qamrun Nahar whose telephone number is (571) 272-3730. The examiner can normally be reached on Mondays through Fridays from 10:00 AM to 6:30 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y Zhen, can be reached on (571) 272-3708. The fax phone number for the organization where this application or processing is assigned is (571) 273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Qamrun Nahar/
Qamrun Nahar
Primary Examiner, AU 2191
Art Unit 2191